SESSION # 2

DIGITAL IMAGE PROCESSING & MACHINE VISION

HOW MACHINES SEE IMAGES : THE DIFFERENCE

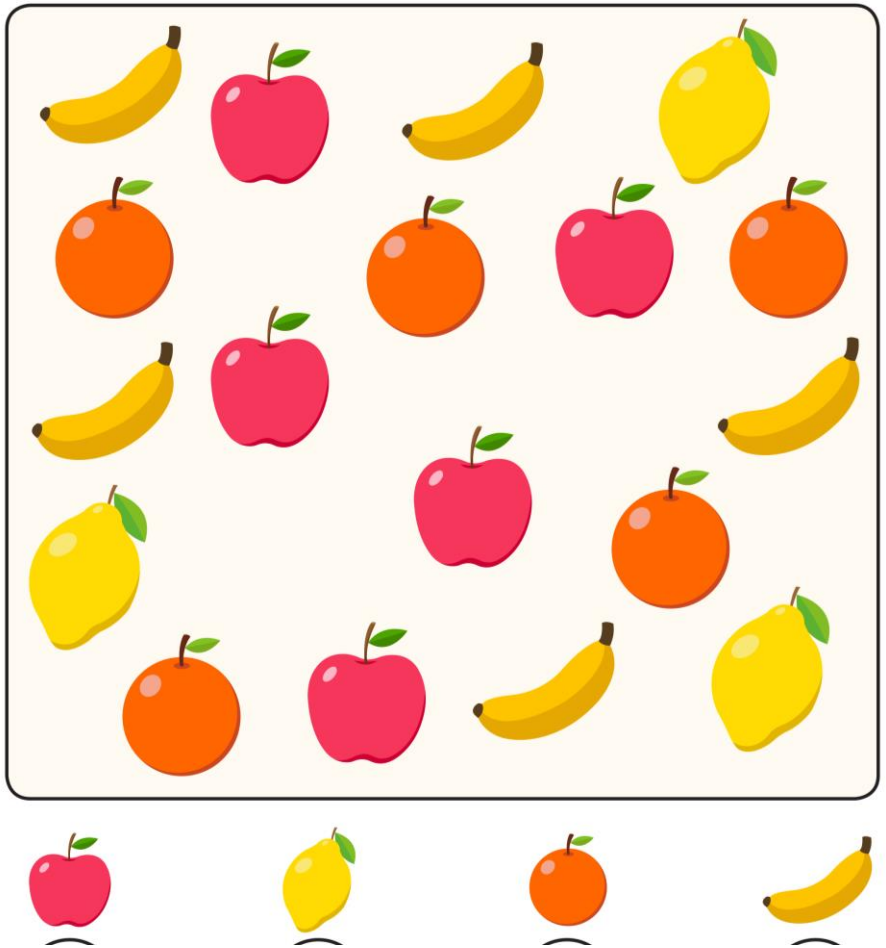
Machines see images in a way fundamentally different from humans. Rather than perceiving a scene as a whole, they break it down into data—grids of pixels, each pixel encoded with values representing color and intensity. This raw data is then processed by algorithms that identify patterns, edges, textures, and objects. Advanced systems, such as those powered by neural networks, use layers of computation to extract higher-level features, like identifying faces, categorizing objects, or interpreting scenes.

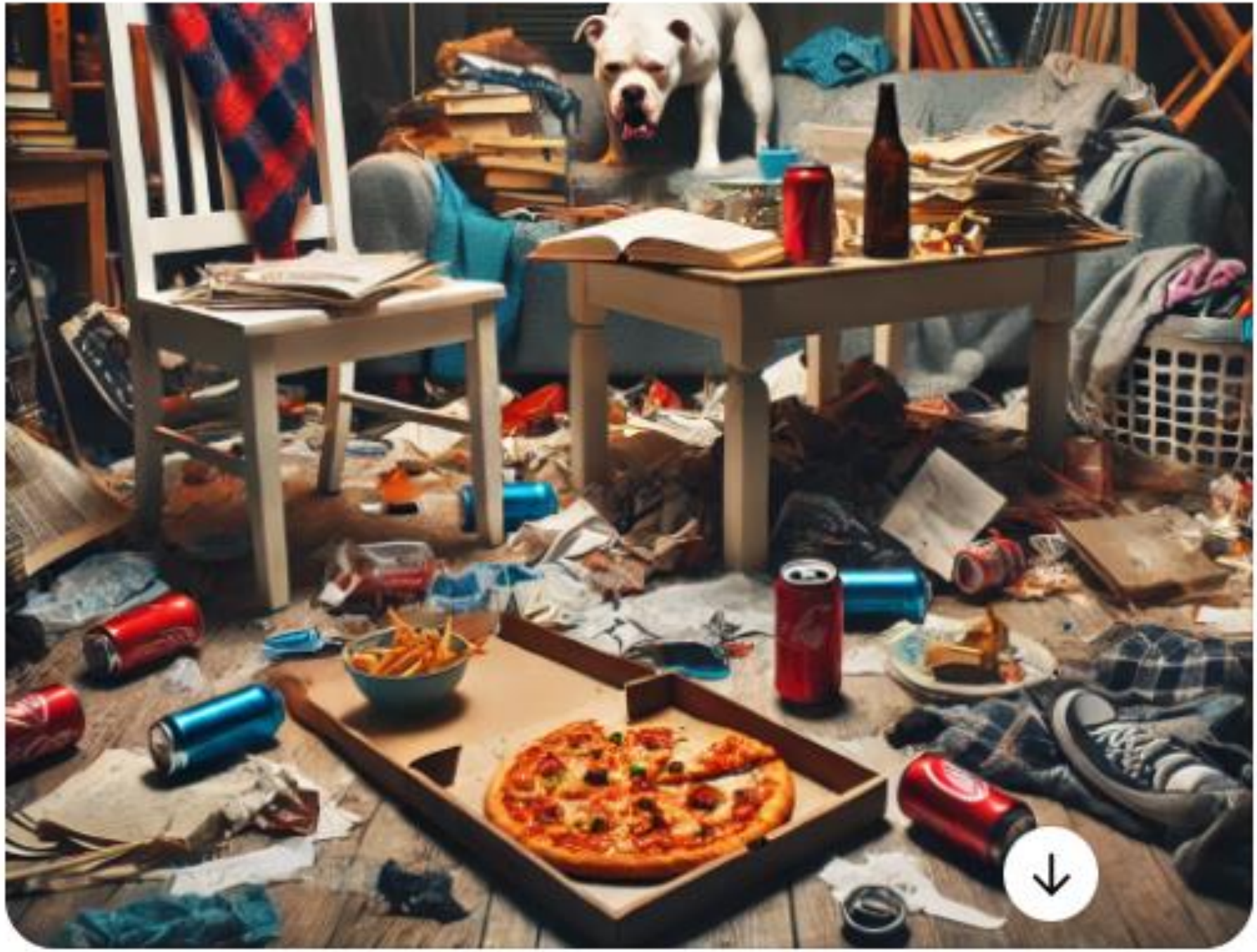
For example, when a machine views a forest image, it identifies individual components—trees, ground, sky—and assigns labels based on patterns it has learned from analyzing millions of similar images. The machine doesn't "see" the beauty or emotion of the forest; it understands it as a structured data

OBJECTS IN AN IMAGE

IDENTIFY NUMBER OF OBJECTS AND CLASSES

below.





Quiz # 01

Count the number of objects and classes in the given image

WHY MACHINES ARE NEEDED TO SEE/PROCESS IMAGES

Healthcare: Machines analyze medical images, such as X-rays or MRIs, to detect abnormalities like tumors or fractures, often with **greater precision than human experts**.

Autonomous Vehicles: Self-driving cars rely on image recognition to detect pedestrians, other vehicles, and road signs, **ensuring safe navigation**.

Security and Surveillance: Facial recognition systems identify individuals in real-time, **enhancing automated security in sensitive areas.**

Environmental Monitoring: Machines analyze satellite imagery to track deforestation, monitor climate changes, or predict natural disasters.

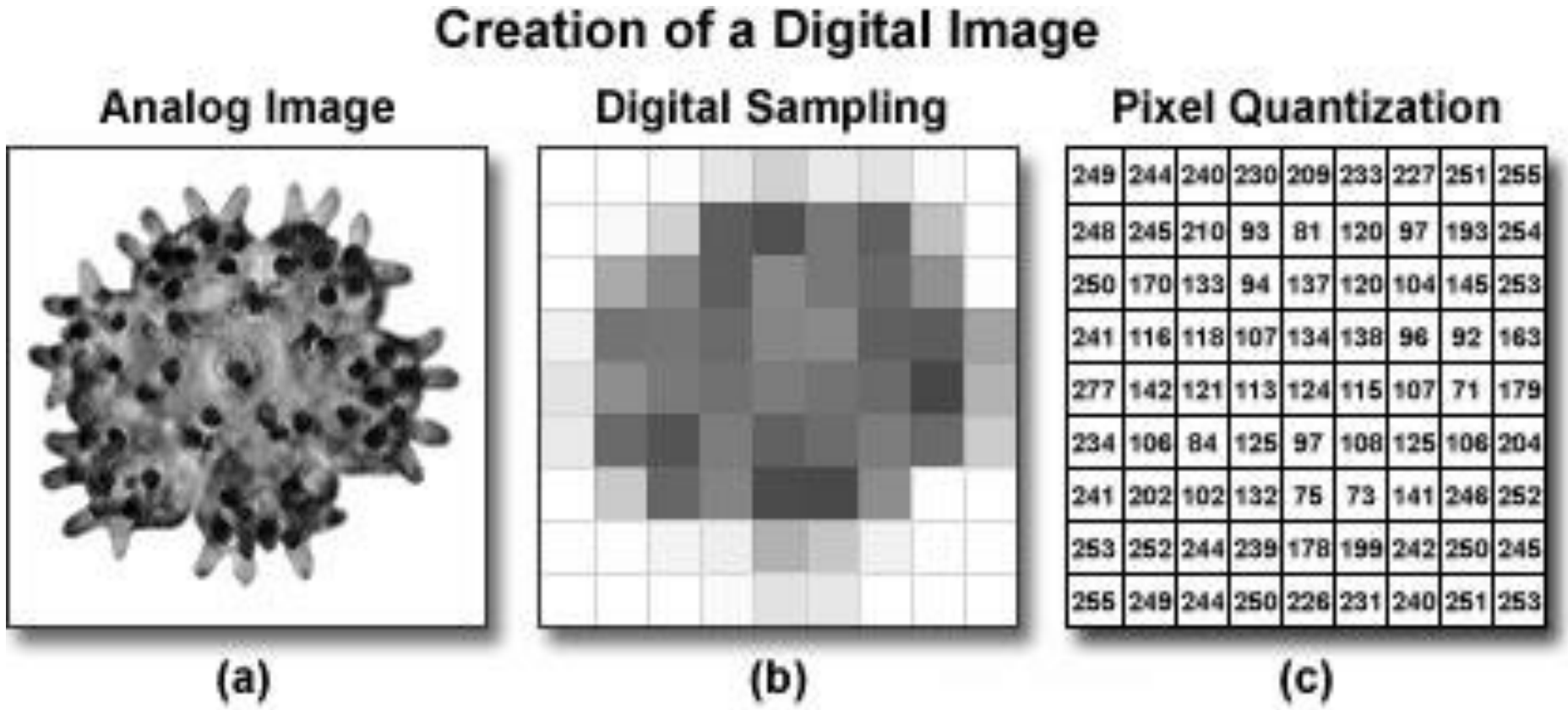
Accessibility: Image recognition enables visually impaired individuals to interpret their surroundings through audio descriptions.

Creative Industries: Machines generate art, enhance photographs, or assist in design by analyzing and recreating visual styles.

Automation
Speed
Accuracy
Consistency
Person in-variant

Functional Parameter	Human Vision	Machine Vision
Adaptability	More adaptable to environmental conditions	Not much adaptable to changing world
Decision Making	Humans are good in making relative comparisons	Machine needs fixed numerical values to decide
Consistency	Human are tired and less consistent	Machines are consistent
Accuracy	Accuracy is subjective	Accuracy is higher
Speed	Human brain is fast in processing	Machines with state of art have limited speed incomparable to human brain
Spectrum	Human can make use of only visible light (390-790mm)	Machines can operate in X-ray and infra red ranges

SAMPLING & QUANTIZATION



QUANTIZATION & RESOLUTION

- The size of the 2-D pixel grid together with the data size stored for each individual image pixel determines the spatial resolution and color quantization of the image.
- The representational power (or size) of an image is defined by its resolution.

QUANTIZATION & RESOLUTION

Aspect	Resolution	Quantization
Definition	Number of pixels in the image	Reducing the range of colors/values in the image
Impact	Affects the detail and sharpness	Affects the color depth and smoothness of gradients
Measurement	Pixels (e.g., 1920×1080)	Bit depth (e.g., 8-bit, 24-bit)
Focus Area	Spatial detail	Color/detail reduction

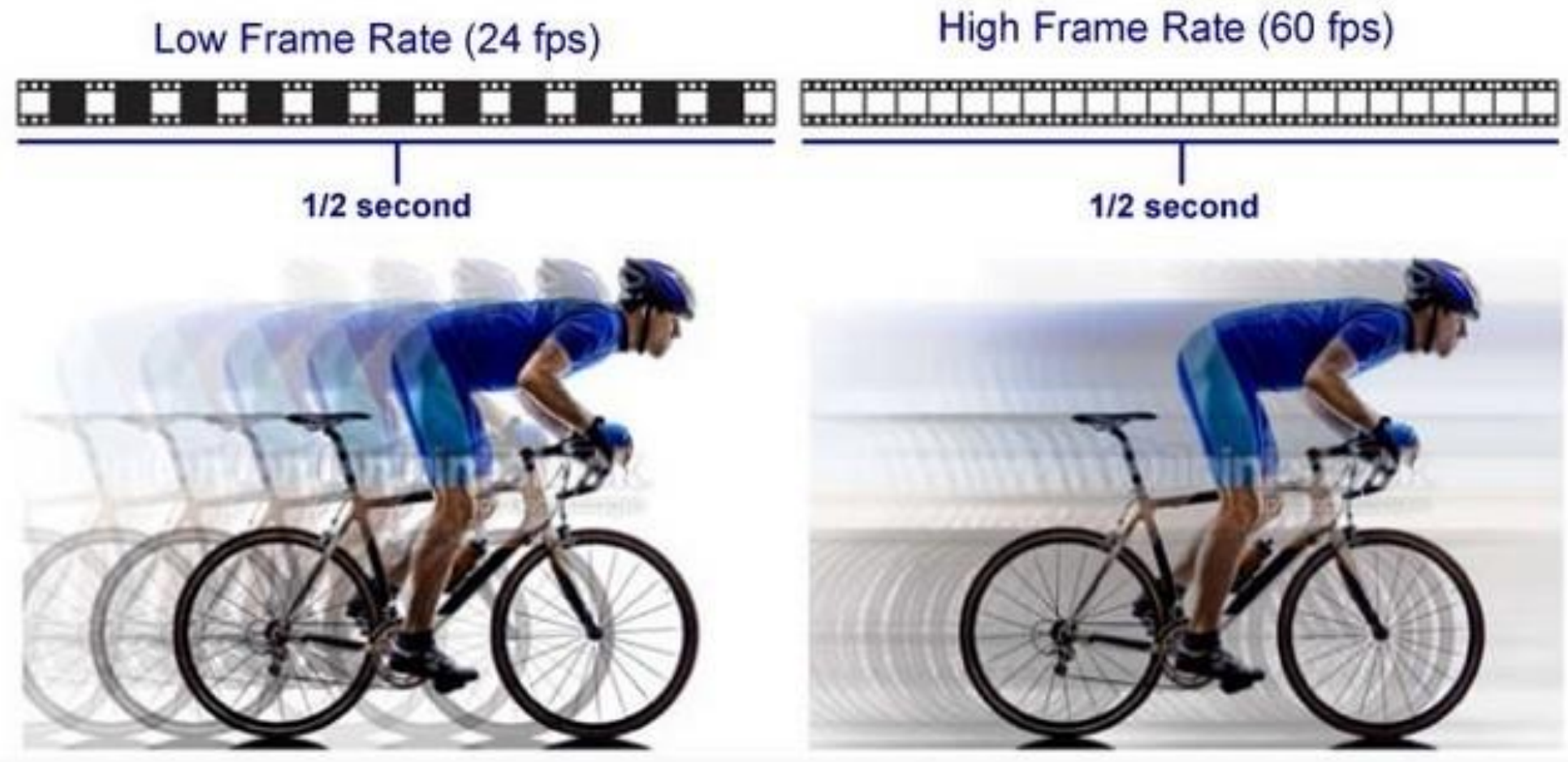
RESOLUTION

- The resolution of an image source (e.g. a camera) can be specified in terms of three quantities; spatial, temporal, and bit resolution
  - **Spatial resolution:**
    - The column (C) by row (R) dimensions of the image define the number of pixels used to cover the visual space captured by the image.
    - This relates to the sampling of the image signal and is sometimes referred to as the pixel or digital resolution of the image. It is commonly quoted as CxR (e.g. 640x480, 800x600, 1024x768, etc.).

RESOLUTION

- **Temporal resolution**
  - For a continuous capture system such as video, this is the number of images captured in a given time period.
  - It is commonly quoted in frames per second (fps), where each individual image is referred to as a video frame (e.g. commonly broadcast TV operates at 25 fps; 25–30 fps is suitable for most visual surveillance; higher frame-rate cameras are available for specialist science/engineering capture).

TEMPORAL RESOLUTION



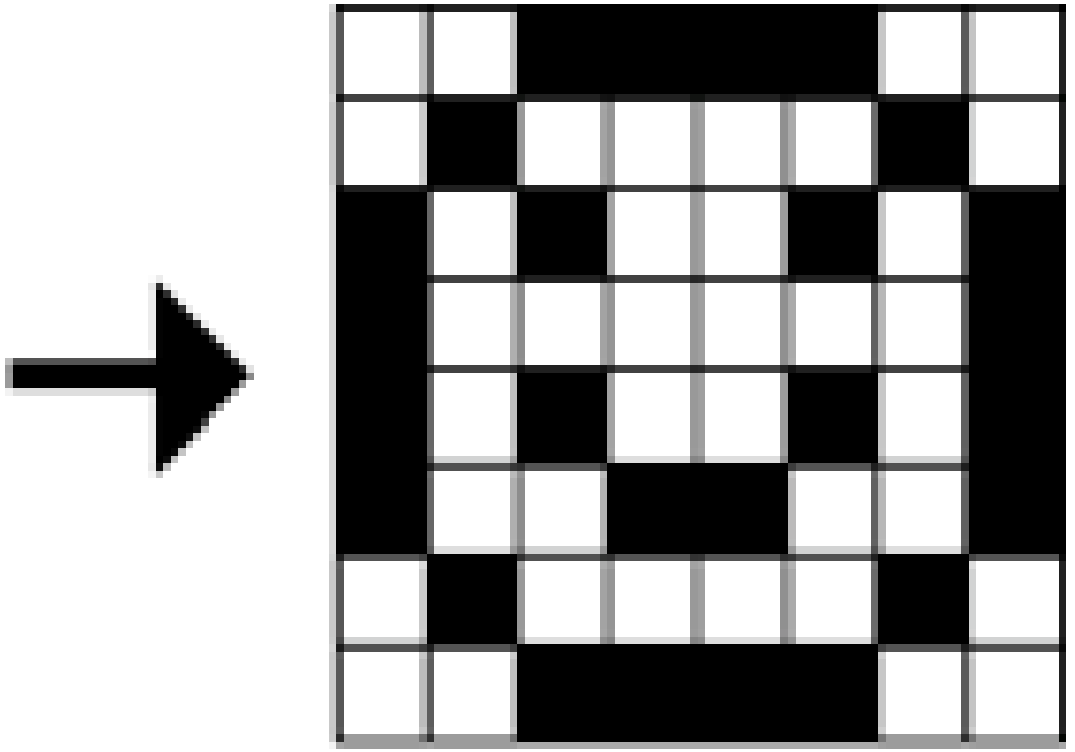
RESOLUTION

Bit resolution

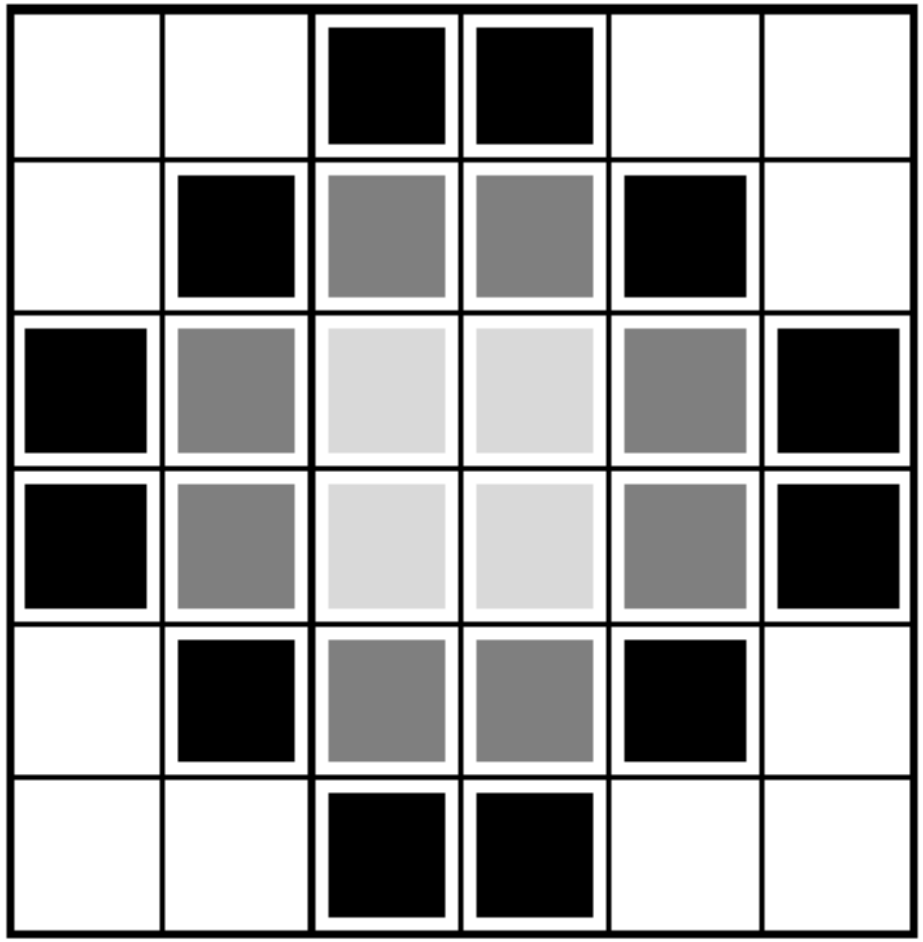
- This defines the number of possible intensity/color values that a pixel may have and relates to the quantization of the image information. For instance a binary image has just two colors (black or white), a grey-scale image commonly has 256 different grey levels ranging from black to white whilst for a color image it depends on the color range in use.
- The bit resolution is commonly quoted as the number of binary bits required for storage at a given quantization level, e.g. binary is 2 bit, grey-scale is 8 bit and color (most commonly) is 24 bit.
- The range of values a pixel may take is often referred to as the dynamic range of an image.

BINARY IMAGE

11000011
10111101
01011010
01111110
01011010
01100110
10111101
11000011

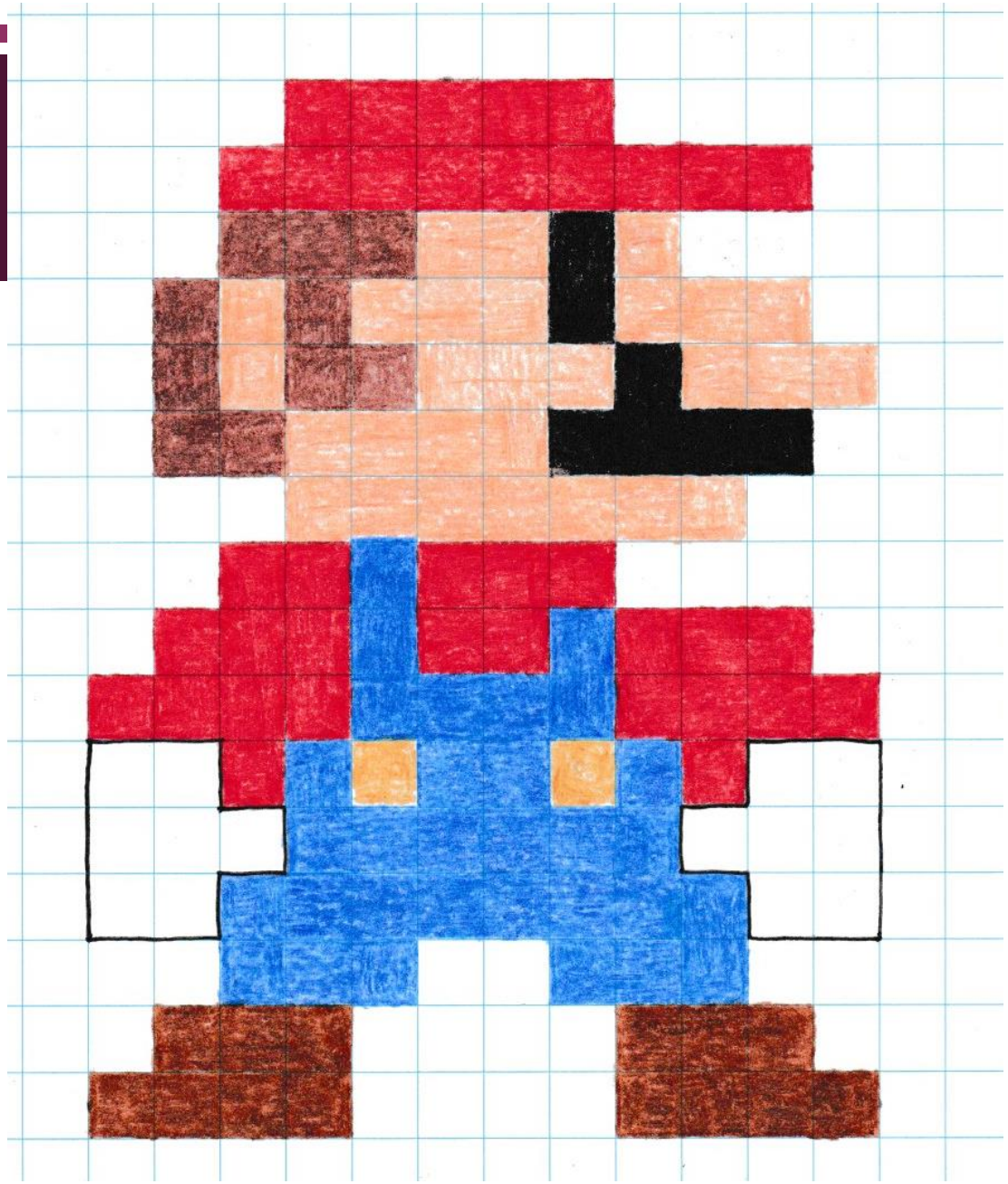


IDENTIFY BIT SIZE?

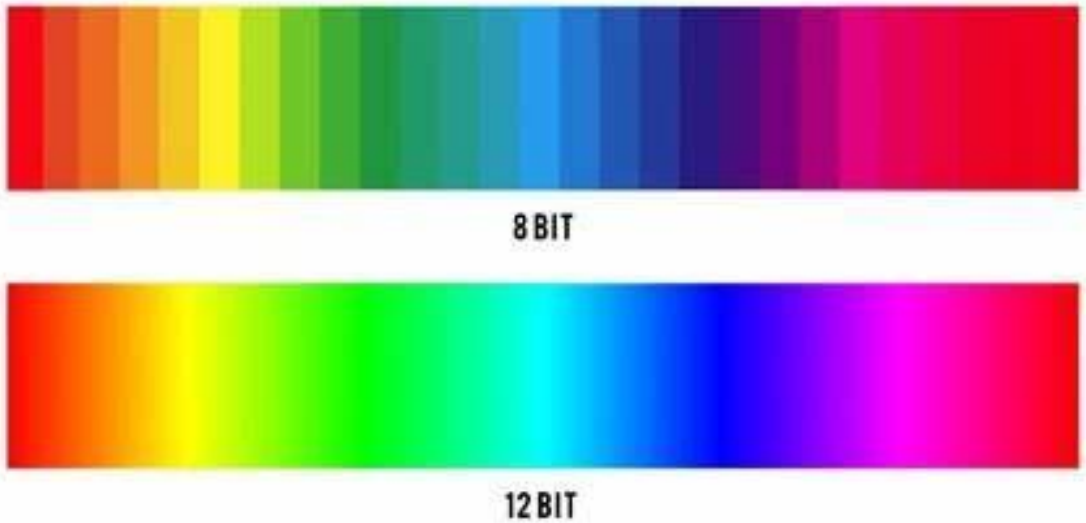


```
11 11 00 00 11 11
11 00 01 01 00 11
00 01 10 10 01 00
00 01 10 10 01 00
11 00 01 01 00 11
11 11 00 00 11 11
```

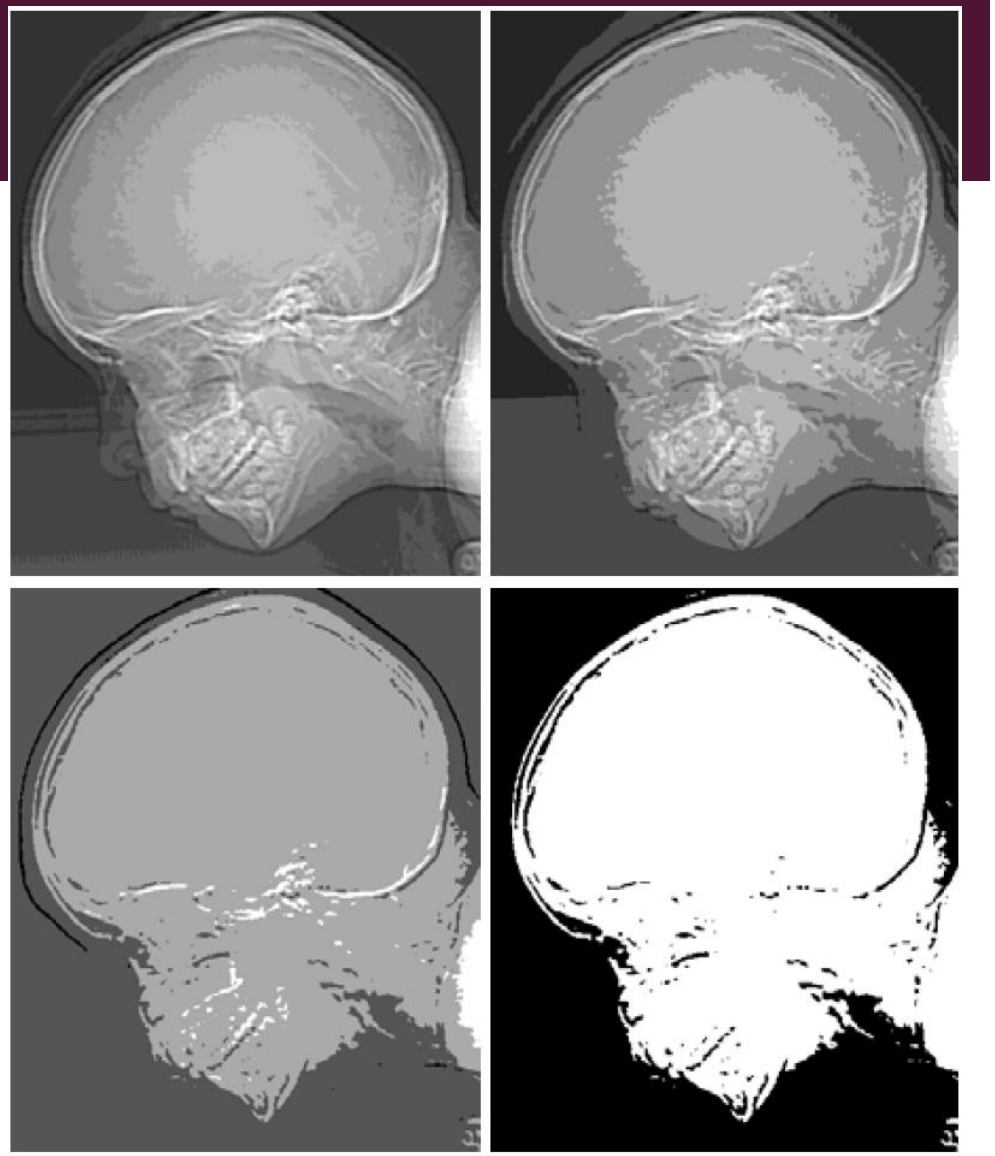
IDENTIFY BIT RESOLUTION?



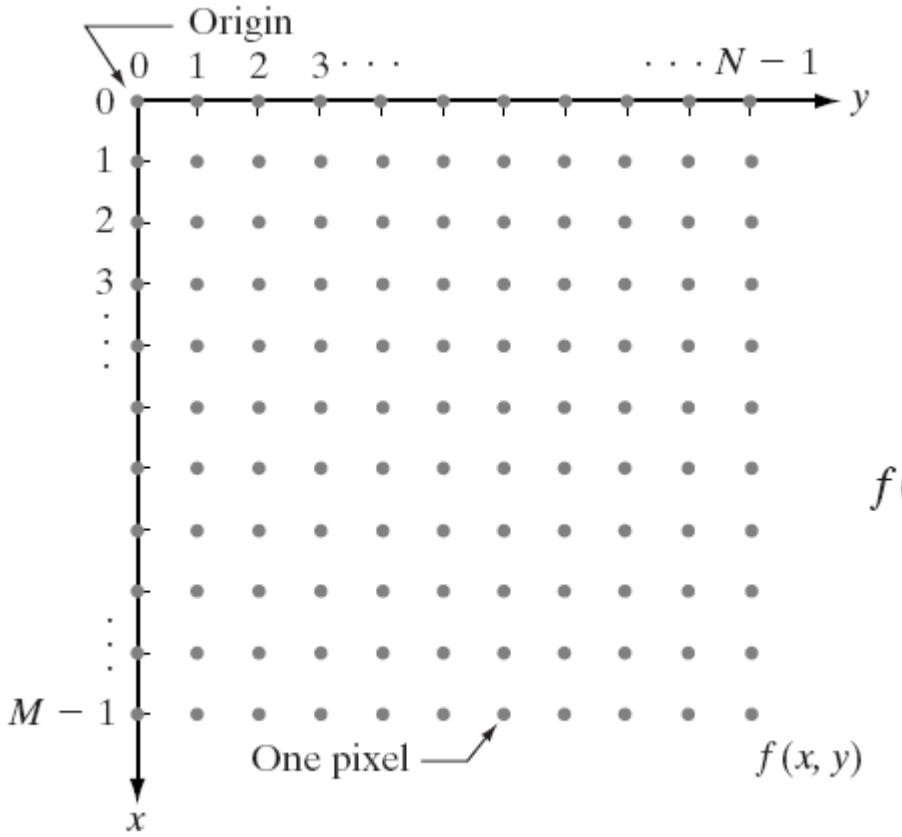
8 BIT VS 12 BIT



GREY SCALE QUANTIZATION

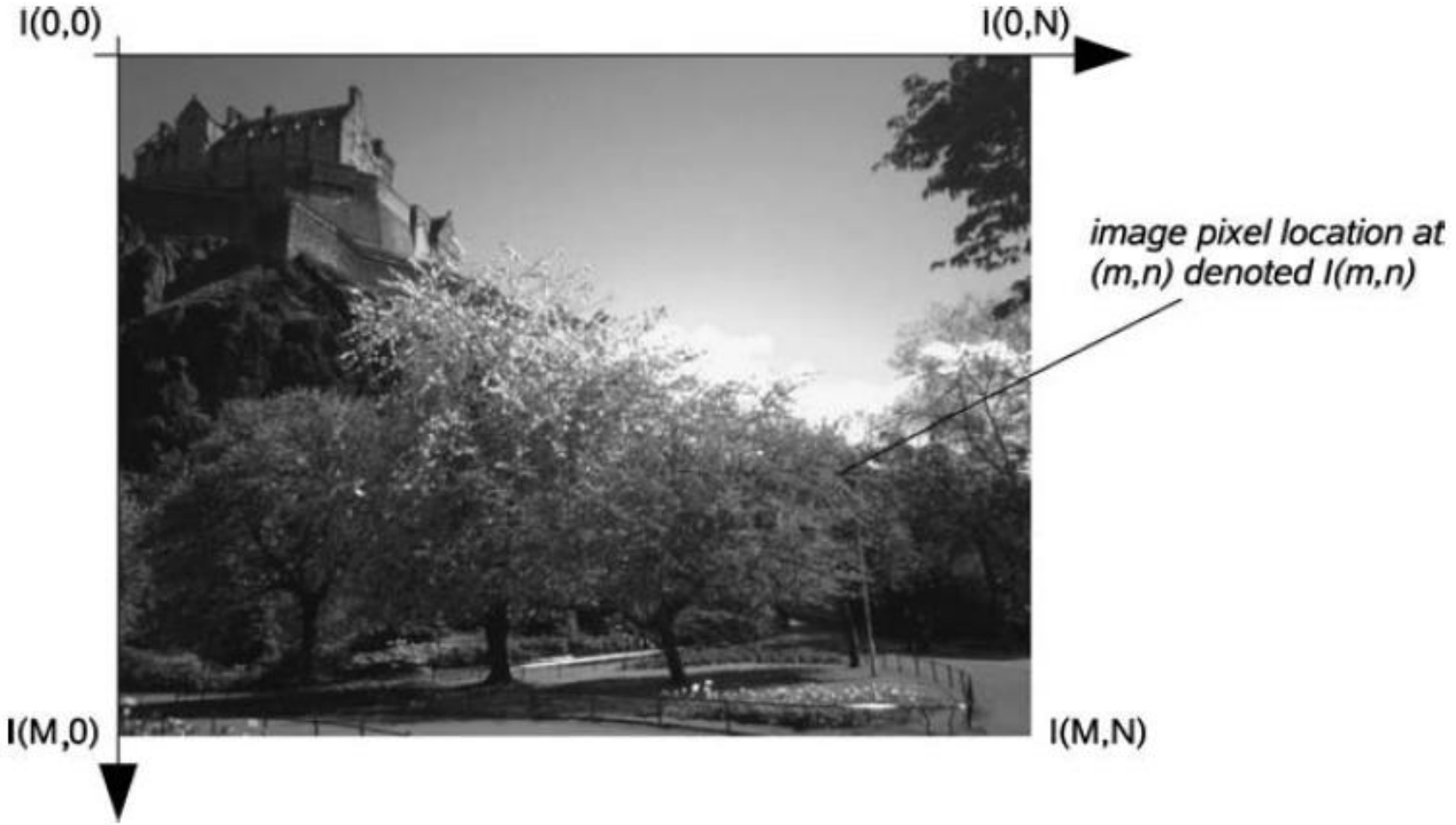


REPRESENTATION OF IMAGE FUNCTION



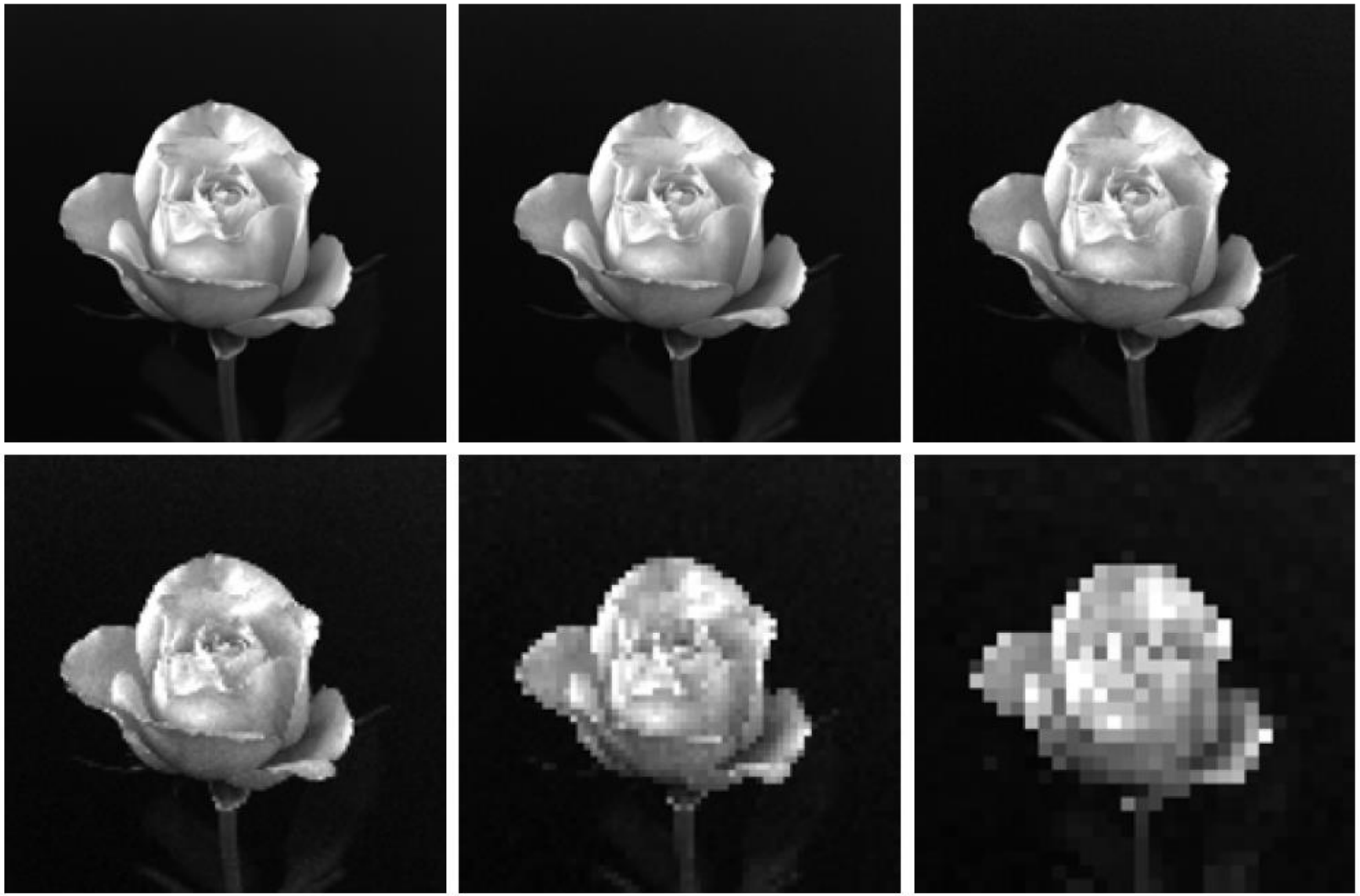
$$f(x, y) = \begin{bmatrix} f(0, 0) & f(0, 1) & \cdots & f(0, N-1) \\ f(1, 0) & f(1, 1) & \cdots & f(1, N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1, 0) & f(M-1, 1) & \cdots & f(M-1, N-1) \end{bmatrix}.$$

REPRESENTATION OF IMAGE FUNCTION



SUBSAMPLING WITH FIXED GRAY LEVELS

FIGURE 2.19 A 1024 × 1024, 8-bit image subsampled down to size 32 × 32 pixels. The number of allowable gray levels was kept at 256.

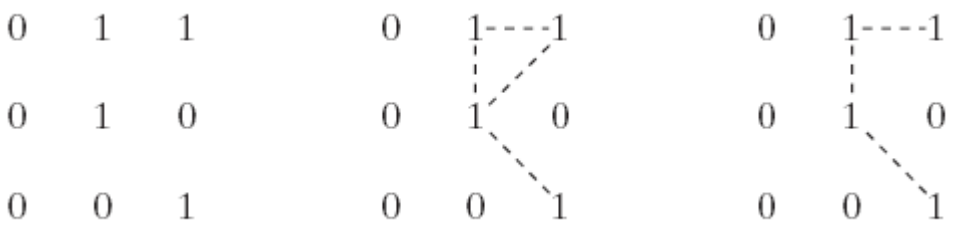


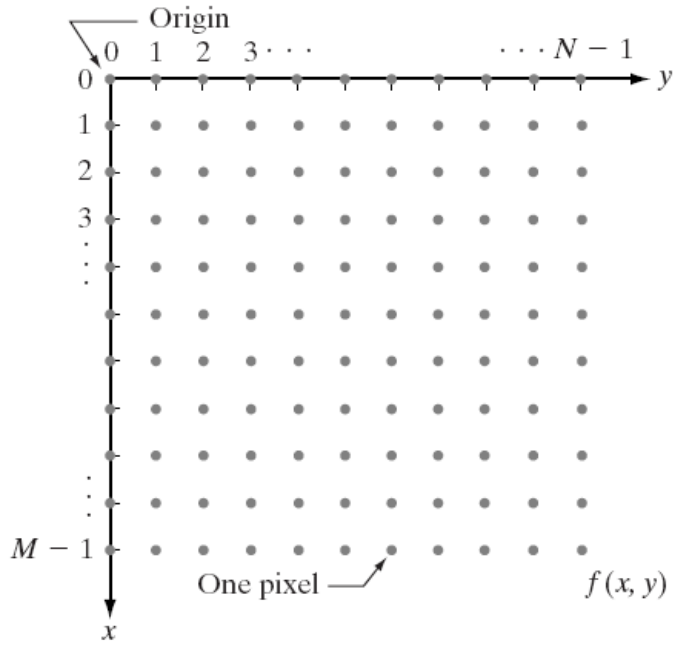
a b c
d e f

FIGURE 2.20 (a) 1024 × 1024, 8-bit image. (b) 512 × 512 image resampled into 1024 × 1024 pixels by row and column duplication. (c) through (f) 256 × 256, 128 × 128, 64 × 64, and 32 × 32 images resampled into 1024 × 1024 pixels.

PIXEL NEIGHBORHOOD

$$(x + 1, y), (x - 1, y), (x, y + 1), (x, y - 1)$$

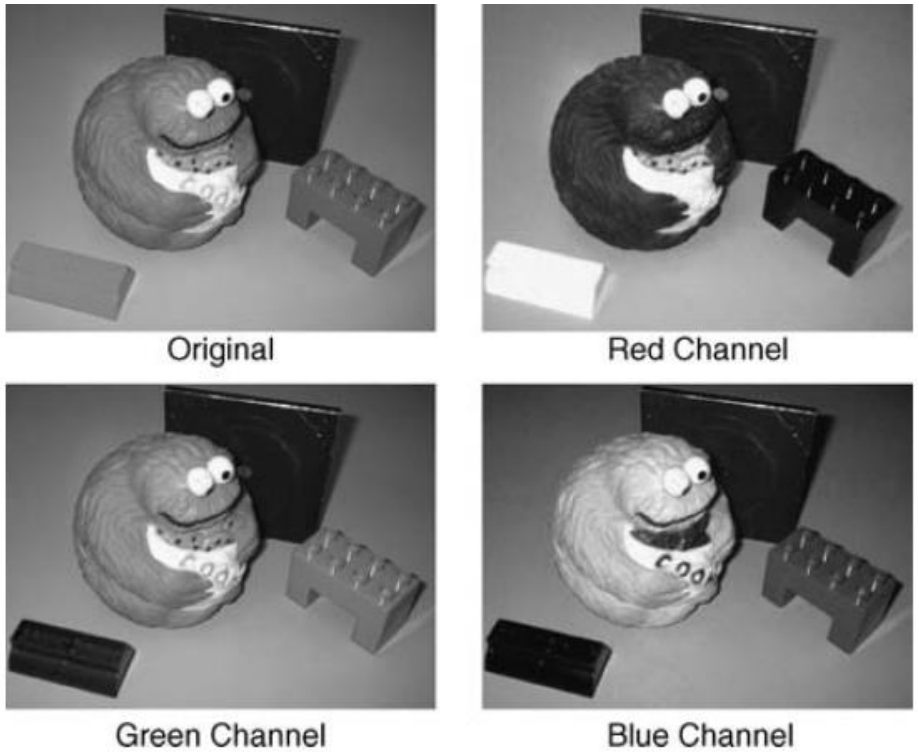




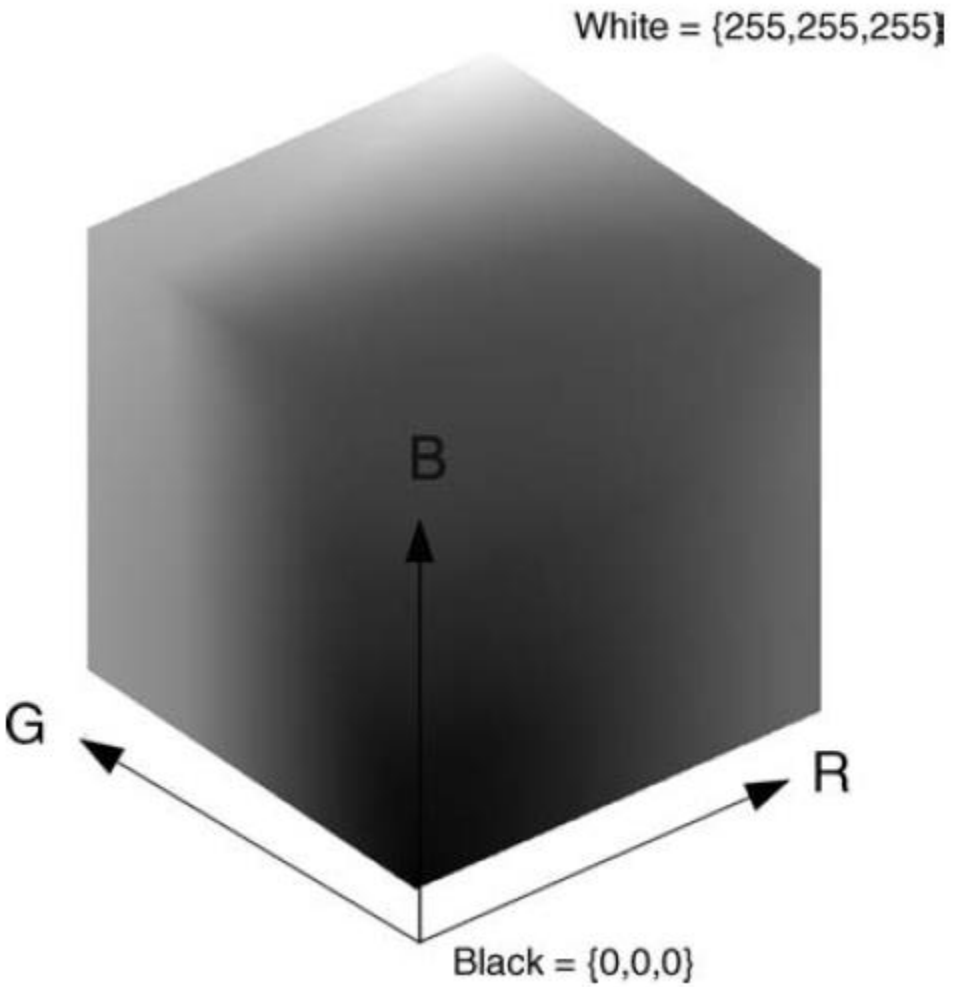
$$(x + 1, y + 1), (x + 1, y - 1), (x - 1, y + 1), (x - 1, y - 1)$$

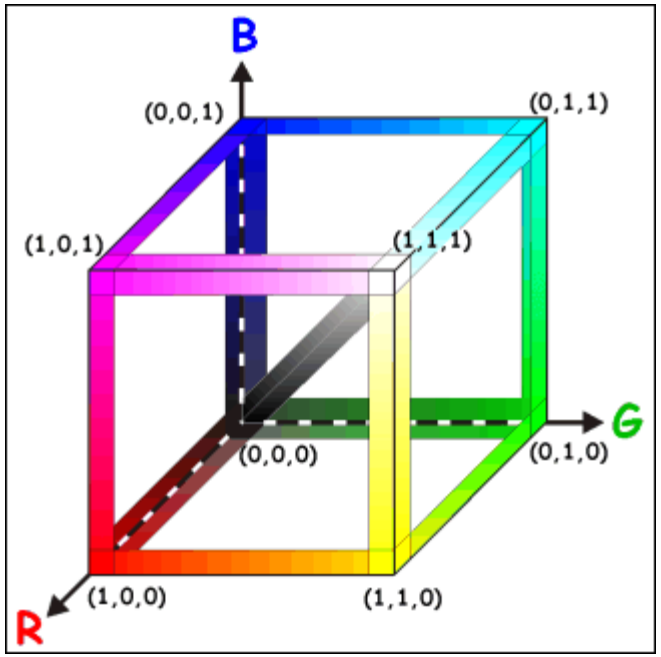
COLOR SPACES

- The representation we use to store the colors, specifying the number and nature of the color channels, is generally known as the color space.



RGB


White = {255,255,255}
B
G
R
Black = {0,0,0}


B
(0,0,1)
(0,1,1)
(1,0,1)
(1,1,1)
G
(0,0,0)
(0,1,0)
R
(1,0,0)
(1,1,0)

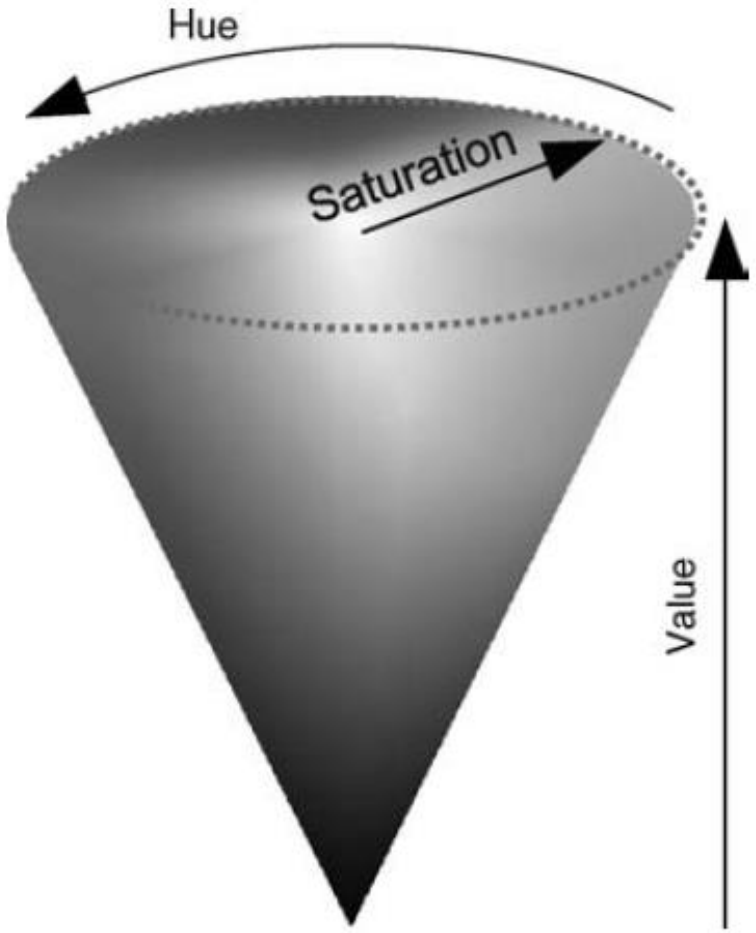
RGB TO GREYSCALE

$$I_{\text{grey-scale}}(n,m) = \alpha I_{\text{colour}}(n,m,r) + \beta I_{\text{colour}}(n,m,g) + \gamma I_{\text{colour}}(n,m,b)$$

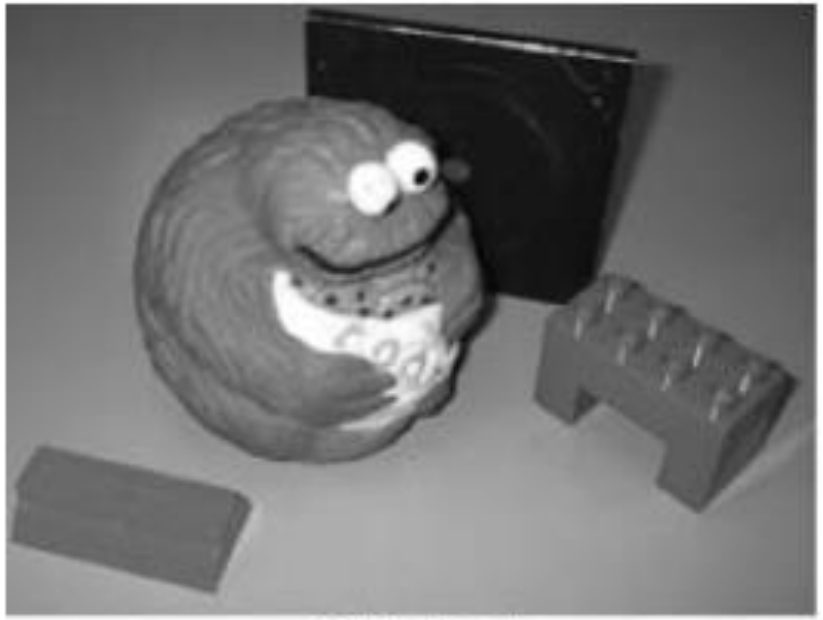
I = rgb2gray(RGB); figure imshow(I)

PERCEPTUAL COLOR SPACE

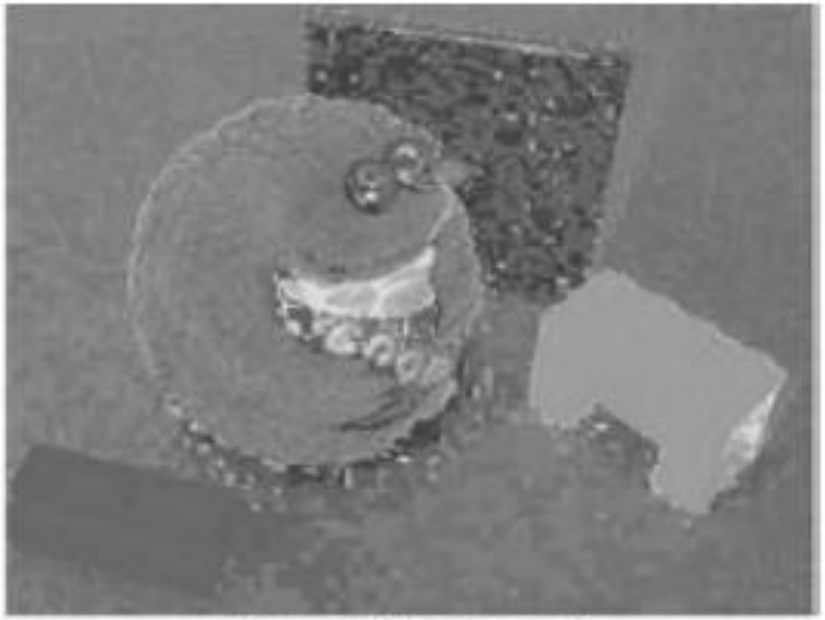
- Hue-Saturation-Value Model



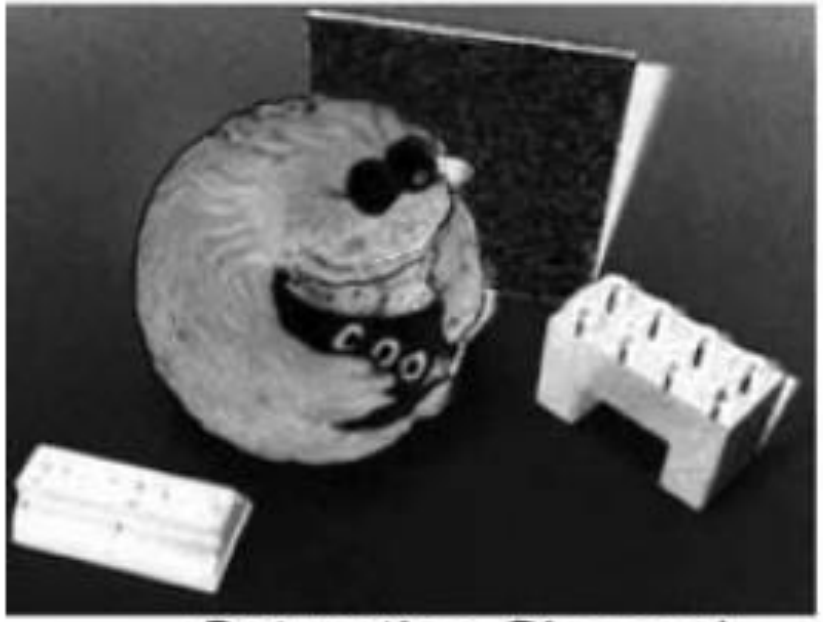
HSV (HIS) MODEL



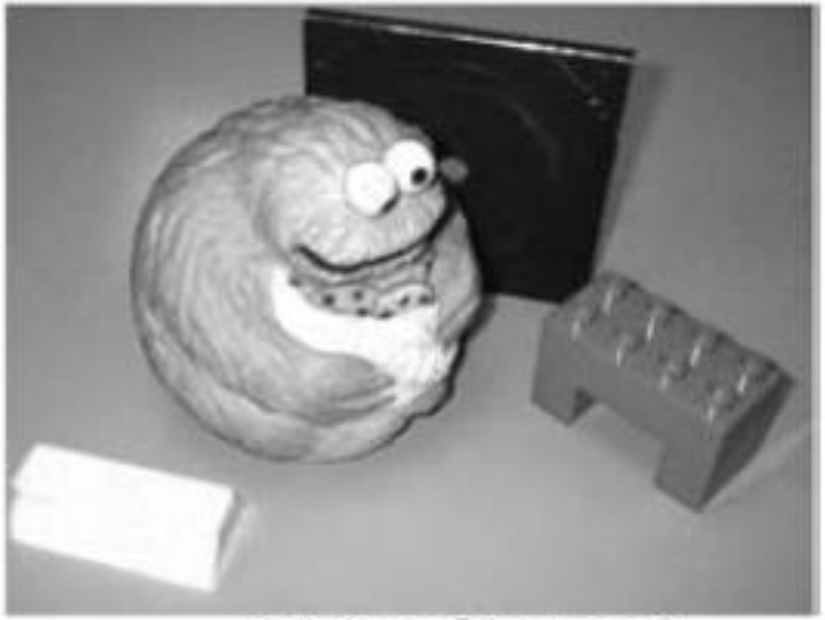
Original



Hue Channel

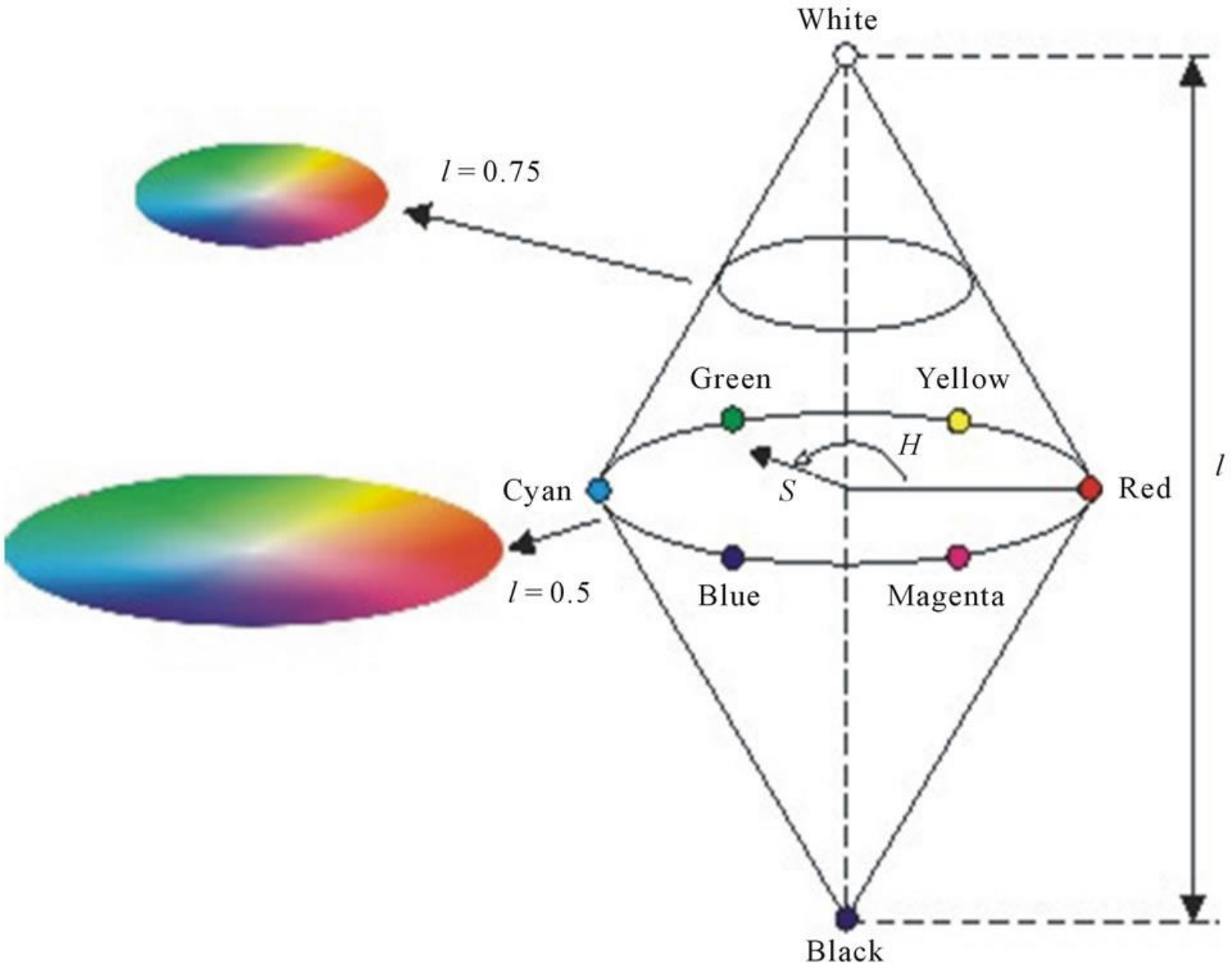


Saturation Channel



Value Channel

HSV TO COLOR SPACE



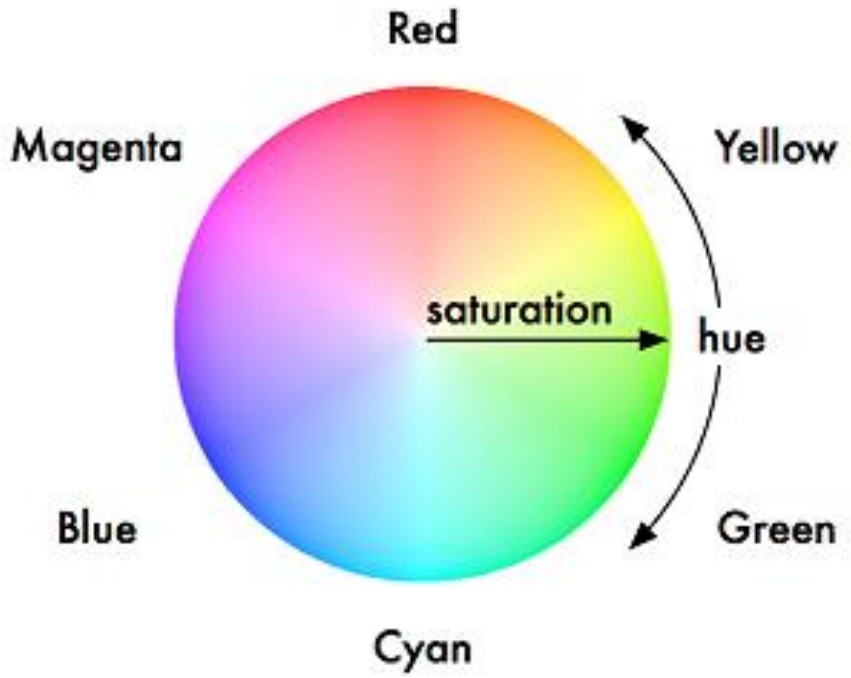
HSV TO COLOR SPACE

$$I = \frac{1}{3}(R + G + B)$$

$$S = 1 - \frac{3}{R + G + B}[\min(R, G, B)]$$

$$H = \cos^{-1}\left[\frac{\frac{1}{2}[(R - G) + (R - B)]}{\sqrt{(R - G)^2 + (R - B)(G - B)}}\right]$$

If B is greater than G, then $H = 360^0 - H$.



ASSIGNMENT # 2

- Apply openCV codes on selected pictures/images and make a presentation of results
- Codes are given here:

Example 1.1

Matlab code	What is happening?
imfinfo('cameraman.tif')	%Query the cameraman image that %is available with Matlab %imfinfo provides information %ColorType is gray scale, width is 256 . . . etc.
I1=imread('cameraman.tif');	%Read in the TIF format cameraman image
imwrite(I1,'cameraman.jpg','jpg');	%Write the resulting array I1 to %disk as a JPEG image
imfinfo('cameraman.jpg')	%Query the resulting disk image %Note changes in storage size, etc.

Comments

- Matlab functions: ***imread***, ***imwrite*** and ***iminfo***.
- Note the change in file size when the image is stored as a JPEG image. This is due to the (lossy) compression used by the JPEG image format.

Example 1.2

Matlab code	What is happening?
A=imread('cameraman.tif');	%Read in intensity image
imshow(A);	%First display image using imshow
imagesc(A);	%Next display image using imagesc
axis image;	%Correct aspect ratio of displayed image
axis off;	%Turn off the axis labelling
colormap(gray);	%Display intensity image in grey scale

Comments

- Matlab functions: ***imshow***, ***imagesc*** and ***colormap***.
- Note additional steps required when using ***imagesc*** to display conventional images.

Example 1.3

Matlab code	What is happening?
B=rand(256).*1000;	%Generate random image array in range 0 1000
imshow(B);	%Poor contrast results using imshow because data %exceeds expected range

imagesc(B);
axis image; axis off;
colormap(gray); colorbar;
	%imagesc automatically scales colourmap to data
	%range

imshow(B,[0 1000]);	%But if we specify range of data explicitly then %imshow also displays correct image contrast

Comments

- Note the automatic display scaling of ***imagesc***.

Example 1.4

Matlab code	What is happening?
B=imread('cell.tif');	%Read in 8 bit intensity image of cell
C=imread('spine.tif');	%Read in 8 bit intensity image of spine
D=imread('onion.png');	%Read in 8 bit colour image
subplot(3,1,1); imagesc(B); axis image;	%Creates a 3 × 1 mosaic of plots
axis off; colormap(gray);	%and display first image
subplot(3,1,2); imagesc(C); axis image; axis off; colormap(jet);	%Display second image %Set colourmap to jet (false colour)
subplot(3,1,3); imshow(D);	%Display third (colour) image

Comments

- Note the specification of different colour maps using ***imagesc*** and the combined display using both ***imagesc*** and ***imshow***.

Example 1.5

Matlab code	What is happening?
B=imread('cell.tif');	%Read in 8 bit intensity image of cell
imview(B);	%Examine grey scale image in interactive viewer
D=imread('onion.png');	%Read in 8 bit colour image.
imview(B);	%Examine RGB image in interactive viewer
B(25,50)	%Print pixel value at location (25,50)
B(25,50)=255;	%Set pixel value at (25,50) to white
imshow(B);	%View resulting changes in image
D(25,50,:)	%Print RGB pixel value at location (25,50)
D(25,50, 1)	%Print only the red value at (25,50)
D(25,50,:)=(255, 255, 255);	%Set pixel value to RGB white
imshow(D);	%View resulting changes in image

Comments

- Matlab functions: ***imview***.
- Note how we can access individual pixel values within the image and change their value.

Example 1.6

Matlab code	What is happening?
D=imread('onion.png');	%Read in 8 bit RGB colour image
Dgray=rgb2gray(D);	%Convert it to a grey scale image
subplot(2,1,1); imshow(D); axis image; subplot(2,1,2); imshow(Dgray);	%Display both side by side

Comments

- Matlab functions: ***rgb2gray***.
- Note how the resulting grayscale image array is 2 D while the originating colour image array was 3 D.

Example 1.7

Matlab code	What is happening?
D=imread('onion.png');	%Read in 8 bit RGB colour image.
Dred=D(:,:,1);	%Extract red channel (first channel)
Dgreen=D(:,:,2);	%Extract green channel (second channel)
Dblue=D(:,:,3);	%Extract blue channel (third channel)
subplot(2,2,1); imshow(D); axis image;	%Display all in 2 × 2 plot
subplot(2,2,2); imshow(Dred); title('red');	%Display and label
subplot(2,2,3); imshow(Dgreen); title('green');	
subplot(2,2,4); imshow(Dblue); title('blue');	

Comments

- Note how we can access individual channels of an RGB image and extract them as separate images in their own right.